CHAPTER 5
Estimation and Confidence Intervals

Introduction

Estimation involves making an inference on the true value, while the confidence interval provides a range of values that we can be confident contains the true value. For example, suppose you are a teacher and you want to estimate the average height of the students in your school. It is not possible to measure the height of every student, so you take a sample of 30 students and measure their heights. Let us say the average height of your sample is 160 cm and the standard deviation is 10 cm. This average of 160 cm is your point estimate of the average height of all students in your school. However, it should be noted that the 30 students sampled may not be a perfect representation of the entire class, as there may be taller or shorter students who were not included. Therefore, it cannot be definitively concluded that the average height of all students in the class is exactly 160 cm. To ad-dress this uncertainty, a confidence interval can be calculated. A confidence interval is an estimate of the range in which the true population mean, the average height of all students in

the class, is likely to lie. It is based on the sample mean and standard deviation and provides a measure of the uncertainty in the estimate. In this example, a 95% confidence interval was calculated, indicating that there is a 95% probability that the true average height of all students in the class falls between 155 cm and 165 cm.

These concepts from descriptive statistics aid in making informed decisions based on the available data by quantifying uncertainty, understanding variations around an estimate, comparing different estimates, and testing hypotheses.

Structure

In this chapter, we will discuss the following topics:

- Points and interval estimation
- Standard error and margin of error
- Confidence intervals

Objectives

By the end of this chapter, readers will be introduced to the concept of estimation in data analysis and explain how to perform it using different methods. Estimation is the process of inferring unknown population parameters from sample data. There are two types of estimation: point estimation and interval estimation. This chapter will also discuss the types of errors in estimation, and how to measure them. Moreover, this chapter will demonstrate how to construct and interpret various confidence intervals for different scenarios, such as comparing means, proportions, or correlations. Finally, this chapter will show how to use t-tests and p-values to test hypotheses about population parameters based on confidence intervals. Examples and exercises will be provided throughout the

chapter to help the reader understand and apply the concepts and methods of estimation.

Point and interval estimate

Point estimate is a single value that represents our best approximate value for an unknown population parameter. It is like taking a snapshot of a population based on a limited sample. This snapshot is not the perfect representation of the entire population, but it serves as a best guess or estimate. Some common point estimates used in statistics are mean, median, mode, variance standard deviation, proportion of the sample. For example, a manufacturing company might want to estimate the average life span of a product. They sample a few products from a production batch and measure their durability. The average lifespan of these samples is a point estimate of the expected lifespan of the product in general.

Tutorial 5.1: An illustration of point estimate based on life span of ten products, is as follows:

```
import numpy as np
# Simulate product lifespans for a sample of 10 products
product_lifespans = [539.84,458.10,474.71,560.67,465.95,474.46,545.27,419.74,447.93,471.52]
# Print the lifespan of the product
print("Lifespan of the product:", product_lifespans)
# Calculate the average lifespan of the sample
average_lifespan = np.mean(product_lifespans)
# Print the point estimate for the average lifespan of the product
print(f"Point estimate for the average lifespan of the product:{average_lifespan:.2f}")
```

Output:

```
Lifespan of the product: [539.84, 458.1, 474.71, 560.67,
465.95, 474.46, 545.27, 419.74, 447.93, 471.52]
Point estimate for the average lifespan of the product:485.82
```

Another example is, you are a salesperson for a grocery store chain and you want to estimate the average household spending on groceries in Oslo. It is impossible to collect data from every household, so you randomly select 500 households and record their food expenditures. The average expenditure of this sample represents the point estimate of the total expenditure of all households in Oslo.

Tutorial 5.2: An illustration of the point estimate based on household spending on groceries, is as follows:

```
import numpy as np
# Set the seed for reproducibility
np.random.seed(0)
# Assume the average household spending on groceries is between $100 and $500
expenditures = np.random.uniform(low=100, high=500, size=500)
# Calculate the point estimate (average expenditure of the sample)
point_estimate = np.mean(expenditures)
print(f"Point estimate of the total expenditure of all households in the Oslo: NOK {point_estimate:.2f}")
```

Output:

```
Point estimate of the total expenditure of all households in the Oslo: NOK 298.64
```

Tutorial 5.3: An illustration of the point estimate based on mean, median, mode, variance standard deviation, proportion of the sample, is as follows:

```
import numpy as np
```

```
# Sample data for household spending on groceries
household_spending = np.array([250.32, 195.87, 228.24, 212.81,
233.99, 241.45, 253.34, 208.53, 231.23, 221.28])
# Calculate point estimate for household spending using mean
mean_household_spending = np.mean(household_spending)
print(f"Point estimate of household spending using mean:{mean_household_spending}")
# Calculate point estimate for household spending using median
median_household_spending = np.median(household_spending)
print(f"Point estimate of household spending using median:{median_household_spending}")
# Calculate point estimate for household spending using mode
mode_household_spending = np.argmax(np.histogram(household_spending)[0])
print(f"Point estimate of household spending using mode:{household_spending[mode_household_spending]}")
# Calculate point estimate for household spending using variance
variance_household_spending = np.var(household_spending)
print(f"Point estimate of household spending using variance:{variance_household_spending:.2f}")
# Calculate point estimate for household spending using standard deviation
std_dev_household_spending = np.std(household_spending)
print(f"Point estimate of household spending using standard deviation:{std_dev_household_spending:.2f}")
```

```
# Calculate point estimate for proportion of households spending over $213
proportion_household_spending_over_213 = len(household_spending[household_spending > 213]) / len(household_spending)
print("Proportion of households spending over NOK 213 :", proportion_household_spending_over_213)
```

Output:

```
Point estimate of household spending using mean:227.706
Point estimate of household spending using median:229.735
Point estimate of household spending using mode:228.24
Point estimate of household spending using variance:305.40
Point estimate of household spending using standard deviation:17.48
Proportion of households spending over NOK 213: 0.7
```

An **interval estimate** is a range of values that is likely to contain the true value of a population parameter. It is calculated from sample data and provides more information about the uncertainty of the estimate than a point estimate. For example, suppose you want to estimate the average height of all adult males in Norway. You take a random sample of 100 adult males and find that their average height is 5'10". This is a point estimate of the average height of all adult males in Norway.

However, you know that the average height of a small sample of men is likely to be different from the average height of the entire population. This is due to sampling error. Sampling error is the difference between the sample mean and the population mean. To account for sampling error, you can calculate an interval estimate. An interval

estimate is a range of values that is likely to contain the true average height of all adult males in Norway.

The formula for calculating an interval estimate is: *point estimate ± margin of error*

The margin of error is the amount of sampling error you are willing to accept. A common margin of error is ±1.96 standard deviations from the sample mean. Using this formula, you can calculate that the 95% confidence interval for the average height of all adult males in the Norway, assuming margin of error of 0.68 inches as: 5'10" ± 0.68 inches. This means that you are 95% confident that the true average height of all adult males in the Norway is between 5'9" and 5'11".

Tutorial 5.4: To estimate an interval of average lifespan of the product, is as follows:

```
import numpy as np
# Simulate product lifespans for a sample of 20 products
product_lifespans = np.random.normal(500, 50, 20)
# Print the lifespan of the product
print("Lifespan of the product:", product_lifespans)
# Calculate the sample mean and standard deviation
sample_mean = np.mean(product_lifespans)
sample_std = np.std(product_lifespans)
# Calculate the 95% confidence interval
confidence_level = 0.95
margin_of_error = 1.96 * sample_std / np.sqrt(20)
lower_bound = sample_mean - margin_of_error
upper_bound = sample_mean + margin_of_error
# Print the 95% confidence interval
print(
    "95% confidence interval for the average lifespan of the product:", (lower_bound, upper_bound)
```

```
)
```

Tutorial 5.4 simulates 20 product lifetimes from a normal distribution with a mean of 500 and a standard deviation of 50. It calculates the sample mean and standard deviation of the simulated data, and then determines 95% confidence interval using the sample mean, standard deviation, and confidence level. The confidence interval is a range of values that is likely to contain the true mean lifetime of the product.

Output:

```
Lifespan of the product: [546.83712318 570.6163853
381.52065474 543.20261502 388.01979707
 520.07495275 561.24352821 503.24280532 436.01554
134 470.72843979
 486.91772771 490.88776081 489.85515796 494.50586
103 510.67400245
 439.57131731 487.89900851 575.91305852 480.76772
884 477.80819534]
95% confidence interval for the average lifespan of the
product: (469.90930134271343, 515.7208647778637)
```

Tutorial 5.5: To estimate an interval of average household spending on groceries based on ten sample data, is as follows:

```
import numpy as np
# Sample data for household spending on groceries
household_spending = np.array([250.32, 195.87, 228.24
,
212.81, 233.99, 241.45, 253.34, 208.53, 231.23, 221.28]
)
# Calculate the sample mean and standard deviation
sample_mean = np.mean(household_spending)
sample_std = np.std(household_spending)
# Calculate the 95% confidence interval
```

```
confidence_level = 0.95
margin_of_error = 1.96 * sample_std / np.sqrt(len(household_spending))
lower_bound = sample_mean - margin_of_error
upper_bound = sample_mean + margin_of_error
# Print the 95% confidence interval
print(
    "95% confidence interval for the average household spending:", (lower_bound, upper_bound)
)
```

Tutorial 5.5 initially calculates the sample mean and standard deviation of the household expenditure data. It then uses these values, along with the confidence level, to calculate the 95% confidence interval. This interval represents a range of values that is likely to contain the true average household spending in the population.

Output:

```
95% confidence interval for the average household spending:
(216.87441676204998, 238.53758323795)
```

Standard error and margin of error

Standard error measures the precision of an estimate of a population mean. The smaller the standard error, the more accurate the estimate. The standard error is calculated as the square root of the variance of the sample. It measures the variability in a sample. It is calculated as follows:

Standard Error = Standard Deviation / √(Sample Size)

For example, a researcher wants to estimate the average weight of all adults in Oslo. She randomly selects 100 adults and finds that their average weight is 160 pounds. The sample standard deviation is 15 pounds. Then, the standard error is as follows:

Standard error = 15 pounds / √(100) pounds = 1.5 pounds

Tutorial 5.6: An implementation of standard error, is as follows:

```
import math
# Sample size
n = 100
# Sample mean
mean = 160
# Sample standard deviation
sd = 15
# Standard error
se = sd / math.sqrt(n)
# Print standard error
print("Standard error:", se)
```

Output:

```
Standard error: 1.5
```

Margin of error on the other side measures the uncertainty in a sample statistic, such as the mean or proportion. It is an estimate of the range within which the true population mean is likely to fall with a specified level of confidence. It is calculated by multiplying the standard error by a z-score, which is a value from a standard normal distribution. The z-score is chosen based on the desired confidence level, t-score is used instead of the z-score when the sample size is small (less than 30), is as follows:

*Margin of Error = z-score * Standard Error*

For example, a researcher wants to estimate the average weight of all adults in Oslo with 95% confidence. The z-score for the 95% confidence level is 1.96. Then, the margin of error is as follows:

*Margin of error = 1.96 * 1.5 pounds = 2.94 pounds*

This means that the researcher is 95% confident that the

average weight of all adults in Oslo is between 157.06 pounds and 162.94 pounds.

Tutorial 5.7: An implementation of margin of error, is as follows:

```
import math
# Sample size
n = 100
# Sample mean
mean = 160
# Sample standard deviation
sd = 15
# Z-score for 95% confidence
z_score = 1.96
# Margin of error
moe = z_score * sd / math.sqrt(n)
# Print margin of error
print("Margin of error:", moe)
# Calculate confidence interval
confidence_interval = (mean - moe, mean + moe)
# Print confidence interval
print("Confidence interval:", confidence_interval)
```

Output:

```
Margin of error: 2.94
Confidence interval: (157.06, 162.94)
```

Tutorial 5.8: Calculating the standard error and margin of error for a survey. For example, a political pollster conducted a survey to estimate the proportion of registered voters in a particular district who support a specific candidate. The survey included 100 randomly selected registered voters in the district, and the results showed that 60% of them support the candidate as follows:

```
import numpy as np
# Example data representing survey responses (1 for support, 0 for not support)
data = [1, 0, 1, 1, 0, 1, 0, 0, 1, 1]
sample_mean = 0.6 # Calculate sample mean
# Calculate sample standard deviation
sample_std = np.std([1 if voter == 1 else 0 for voter in [support for support in data]])
# Calculate standard error
standard_error = sample_std / np.sqrt(len(data))
print(f"Standard Error:{standard_error:.2f}")
# Find z-score for 95% confidence level
z_score = 1.96
# Calculate margin of error
margin_of_error = z_score * standard_error
print(f"Margin of Error:{margin_of_error:.2f}")
```

Output:

```
Standard Error:0.09
Margin of Error:0.19
```

A standard error of 0.09 is an indication that your sample mean is relatively close to the population mean. 0.6 is the sample proportion because 60% support the candidate. This means that the pollster can be 95% confident that the true proportion of registered voters in the district who support the candidate is between 41% and 79%.

Confidence intervals

All confidence intervals are interval estimates, but not all interval estimates are confidence intervals. Interval estimate is a broader term that refers to any range of values that is likely to contain the true value of a population parameter. For instance, if you have a population of

students and want to estimate their average height, you might reason that it is likely to fall between 5 feet 2 inches and 6 feet 2 inches. This is an interval estimate, but it does not have a specific probability associated with it.

Confidence interval, on the other hand, is a specific type of interval estimate that is accompanied by a probability statement. For example, a 95% confidence interval means that if you repeatedly draw different samples from the same population, 95% of the time, the true population parameter will fall within the calculated interval.

As discussed, confidence interval is also used to make inferences about the population based on the sample data.

Tutorial 5.9: Suppose you want to estimate the average height of all adult women in your city. You take a sample of 100 women and find that their average height is 5 feet 5 inches. You want to estimate the true average height of all adult women in the city with 95% confidence. This means that you are 95% confident that the true average height is between 5 feet 3 inches and 5 feet 7 inches. Based on this example a Python program illustrating confidence intervals, is as follows:

```
1. import numpy as np
2. from scipy import stats
3. # Sample data
4. data = np.array([5.1, 5.2, 5.3, 5.4, 5.5, 5.6, 5.7, 5.8, 5.9,
    6])
5. # Calculate sample mean and standard deviation
6. mean = np.mean(data)
7. std = np.std(data)
8. # Calculate confidence interval with 95% confidence lev
   el
9. margin_of_error = stats.norm.ppf(0.975) * std / np.sqrt(
   len(data))
```

```
confidence_interval = (mean - margin_of_error, mean + margin_of_error)
print("Sample mean:", mean)
print("Standard deviation:", std)
print("95% confidence interval:", confidence_interval)
```

Output:

```
Sample mean: 5.55
Standard deviation: 0.2872281323269015
95% confidence interval: (5.371977430445669, 5.728022569554331)
```

The sample mean is 5.55, indicating that the average height in the sample is 5.55 feet. The standard deviation is 0.287, indicating that the heights in the sample vary by about 0.287 feet. The 95% confidence interval is (5.371, 5.72), which suggests that we can be 95% confident that the true average height of all adult women in the city falls within this range. To put it simply, if we were to take multiple samples of 10 women from the city and calculate the average height of each sample, the true average height would fall within the range of 5.37 feet to 5.72 feet 95% of the time.

Tutorial 5.10: A Python program to illustrate confidence interval for the age column in the diabetes dataset, is as follows:

```
import pandas as pd
# Load the diabetes data from a csv file
diabities_df = pd.read_csv("/workspaces/ImplementingStatisticsWithPython/data/chapter1/diabetes.csv")
# Calculate the mean and standard deviation of the 'Age' column
mean = diabities_df['Age'].mean()
std_dev = diabities_df['Age'].std()
# Calculate the standard error
```

```
std_err = std_dev / (len(diabities_df['Age']) ** 0.5)
# Calculate the 95% Confidence Interval
ci = stats.norm.interval(0.95, loc=mean, scale=std_err)
print(f"95% confidence interval for the 'Age' column is {ci}")
```

Output:

```
95% confidence interval for the 'Age' column is (32.40915352661263, 34.0726173067207)
```

Types and interpretation

The importance of confidence intervals lies in their ability to measure the uncertainty or variability around a sample estimate. Confidence intervals are especially useful when studying an entire population is not feasible, so researchers select a sample or subgroup of the population.

Following are some common types of confidence intervals:

- A confidence interval for a mean estimates the population mean. It is used especially when the data follows a normal distribution. It is discussed in *Point Interval Estimate*, and *Confidence Interval* above.
- When data does not follow a normal distribution, various methods may be used to calculate the confidence interval. For example, suppose you are researching the duration of website loading times. You have collected data from 20 users and discovered that the load times are not normally distributed, possibly due to a few users having slow internet connections that skew the data. In this scenario, one way to calculate the confidence interval is to use the bootstrap method. To estimate the confidence interval, the data is resampled with replacement multiple times, and the mean is calculated each time. The distribution of these means is then used.

Tutorial 5.11: A Python program that uses the bootstrap

method to calculate the confidence interval for non-normally distributed data, is as follows:

```
import numpy as np
def bootstrap(data, num_samples, confidence_level):
    # Create an array to hold the bootstrap samples
    bootstrap_samples = np.zeros(num_samples)
    # Generate the samples
    for i in range(num_samples):
        sample = np.random.choice(data, len(data), replace=True)
        bootstrap_samples[i] = np.mean(sample)
    # Calculate the confidence interval
    lower_percentile = (1 - confidence_level) / 2 * 100
    upper_percentile = (1 + confidence_level) / 2 * 100
    confidence_interval = np.percentile(
        bootstrap_samples, [lower_percentile, upper_percentile])
    return confidence_interval
# Suppose these are your load times
load_times = [1.2, 0.9, 1.3, 2.1, 1.8, 2.4, 1.9, 2.2, 1.7,
              2.3, 1.5, 2.0, 1.6, 2.5, 1.4, 2.6, 1.1, 2.7, 1.0, 2.8]
# Calculate the confidence interval
confidence_interval = bootstrap(load_times, 1000, 0.95)
print(f"95% confidence interval : {confidence_interval}")
```

Output:

```
95% confidence interval : [1.614875 2.085]
```

- A confidence interval for proportions estimates the population proportion. It is used when dealing with categorical data. More about this is illustrated in *Confidence Interval For Proportion*.
- Another type of confidence interval estimates the difference between two population means or proportions.

It is used when you want to compare the means or proportions of two populations.

Confidence interval and t-test relation

The t-test is used to compare the means of two independent samples or the mean of a sample to a population mean. It is a type of hypothesis test used to determine whether there is a statistically significant difference between the two means. The t-test assumes that the two samples are drawn from normally distributed populations with equal variances. As the confidence interval is calculated using the sample mean, the standard error of the mean, and the desired confidence level.

Tutorial 5.12: To illustrate the use of the t-test for confidence intervals, consider the following example: Suppose we want to estimate the average height of adult male basketball players in the United States. We randomly sample 50 male basketball players and measure their heights. We then calculate the sample mean and standard deviation of the heights as follows:

```
1. import numpy as np
2. # Sample heights of 50 male basketball players
3. heights = np.array([75, 78, 76, 79, 80, 81, 82, 83, 84,
   85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95,
4.                     96, 97, 98, 99, 100, 101, 102, 103,
   104, 105, 106, 107, 108, 109, 110, 111, 112, 113, 114,
5.                     115, 116, 117, 118, 119, 120])
6. # Calculate sample mean and standard deviation
7. mean = heights.mean()
8. std = heights.std()
```

Then, we will now calculate a 95% confidence interval for the mean height of male basketball players in the population. To determine the appropriate standard error of the mean, we will use the t-test since we are working with a

sample of the population as follows:

```
from scipy import stats
# Calculate degrees of freedom
df = len(heights) - 1
# Calculate t-statistic for 95% confidence level
t = stats.t.ppf(0.95, df)
# Calculate standard error of the mean
sem = std / np.sqrt(len(heights))
# Calculate confidence interval
ci = (mean - t * sem, mean + t * sem)
print("95% confidence interval for population mean:", ci
)
```

The *Tutorial 5.12* output shows, we can be 95% confident that the true population mean height of male basketball players in the United States is between 77.6 and 84 inches. Here the t-test is used to calculate the **Standard Error of the Mean** (**SEM**), which is a crucial component of the confidence interval. The SEM represents the average difference between the sample mean and the true population mean. To account for the variability within the sample, the t-test considers the sample size and the standard deviation of the heights. This ensures that the confidence interval is not overly narrow or excessively wide, providing a reliable range for the true population mean.

Confidence interval and p-value

A confidence interval is a range of values that likely contains the true value of a parameter with a certain level of confidence. A p-value is a probability that measures the compatibility of the observed data with a null hypothesis. The relationship between confidence intervals and p-values is based on the same underlying theory and calculations, but they convey different information. A p-value indicates

whether the observed data are statistically significant or not, that is, whether they provide enough evidence to reject the null hypothesis or not. A confidence interval provides information on the precision and uncertainty of an estimate, indicating how close it is to the true value and the degree to which it may vary due to random error. One way to understand the relationship is to think of the confidence interval as arms that embrace values consistent with the data. If the null value, usually zero or one, falls within the confidence interval, it is not rejected by the data, and the p-value must be greater than the significance level, usually 0.05. If the null value falls outside the confidence interval, it is rejected by the data, and the p-value must be less than the significance level.

For example, let us say you wish to test the hypothesis that the average height of Norwegian men is 180 cm. To do so, you randomly select a sample of 100 men and measure their heights. You calculate the sample mean and standard deviation, and then you construct a 95% confidence interval for the population mean as follows:

$$\bar{x} \pm 1.96 \frac{s}{\sqrt{n}}$$

Where $\bar{x}$ sample mean, s is sample standard deviation, and n is sample size.

Assuming a confidence interval of (179.31, 181.29), we can conclude with 95% confidence that the true mean height of men in Norway falls between 179.31 and 181.29 cm. As the null value of 180 is within this interval, we cannot reject the null hypothesis at the 0.05 significance level. The p-value for this test must is 0.5562, indicating that the observed data are not very unlikely under the null hypothesis. On the other hand, if you obtain a confidence interval that does not include 180, such as (176.5, 179.5), it

would mean that you can be 95% confident that the actual mean height of men in Norway falls outside the hypothesized value of 180 cm. As the null value of 180 would lie outside this interval, you would reject the null hypothesis at the 0.05 significance level. The p-value for this test would be less than 0.05, indicating that it is highly improbable for the observed data to be true under the null hypothesis.

Tutorial 5.13: To illustrate the use of the p-value and confidence intervals, is as follows:

```
# import numpy and scipy libraries
import numpy as np
import scipy.stats as st
# set the random seed for reproducibility
np.random.seed(0)
# generate a random sample of 100 heights from a normal distribution
# with mean 180 and standard deviation 5
heights = np.random.normal(180, 5, 100)
# calculate the sample mean and standard deviation
mean = np.mean(heights)
std = np.std(heights, ddof=1)
# calculate the 95% confidence interval for the population mean
# using the formula: mean +/- 1.96 * std / sqrt(n)
n = len(heights)
lower, upper = st.norm.interval(0.95, loc=mean, scale=std/np.sqrt(n))
# print the confidence interval
print(f"95% confidence interval for the population mean : ({lower:.2f}, {upper:.2f})")
# test the null hypothesis that the population mean is 180
```

```
# using a one-sample t-test
t_stat, p_value = st.ttest_1samp(heights, 180)
# print the p-value
print(f"P-value for the one-sample t-test : {p_value:.4f}")
# compare the p-
value with the significance level of 0.05
# and draw the conclusion
if p_value < 0.05:
    print("We reject the null hypothesis that the populati
on mean is 180")
else:
    print("We fail to reject the null hypothesis that the po
pulation mean is 180")
```

Output:

```
95% confidence interval for the population mean : (179.
31, 181.29)
P-value for the one-sample t-test : 0.5562
We fail to reject the null hypothesis that the population
mean is 180
```

This indicates that the confidence interval includes the null value of 180, and the p-value is greater than 0.05. Therefore, we cannot reject the null hypothesis due to insufficient evidence.

Confidence interval for mean

Some of the concepts have already been described and highlighted in the *Types and Interpretation* section above. Let us see the what, how and when of confidence interval for the mean. Confidence interval for the mean is a range of values that, with a certain level of confidence, is likely to contain the true mean of a population. It is best to use this type of confidence interval when we have a sample of numerical data from a population and we want to estimate the average of the population.

For example, let us say you want to estimate the average height of students in a class. You randomly select 10 students and measure their heights in centimeters. You get the following data:

```
heights = [160, 165, 170, 175, 180, 185, 190, 195, 200, 205]
```

To calculate the 95% confidence interval for the population mean, use the **t.interval** function from the **scipy.stats** library. The confidence parameter should be set to 0.95, and the degrees of freedom should be set to the sample size minus one. Additionally, provide the sample mean and the standard error of the mean as arguments.

Tutorial 5.14: An example to compute confidence interval for mean of the average height of students, is as follows:

```
import numpy as np
import scipy.stats as st
mean = np.mean(heights) # sample mean
se = st.sem(heights) # standard error of the mean
df = len(heights) - 1 # degrees of freedom
ci = st.t.interval(confidence=0.95, df=df, loc=mean, scale=se) # confidence interval
print(f"Confidence interval: {ci}")
```

Output:

```
Confidence interval: (171.67074705193303, 193.32925294806697)
```

This indicates a 95% confidence interval for the true mean height of students in the class, which falls between 171.67 and 193.32 cm.

Confidence interval for proportion

A confidence interval for a proportion is a range of values that, with a certain level of confidence, likely contains the true proportion of a population. The use this type of

confidence interval is when we have a sample of categorical data from a population and we want to estimate the percentage of the population that belongs to a certain category. For example, let us say you want to estimate the proportion of students in a class who prefer chocolate ice cream over vanilla ice cream. You randomly select 50 students and ask them about their preference. You get the following data:

```
preferences = ['chocolate', 'vanilla', 'chocolate', 'chocolate', 'vanilla', 'chocolate', 'chocolate', 'vanilla', 'chocolate',
            'vanilla', 'chocolate', 'chocolate', 'vanilla', 'chocolate', 'chocolate', 'vanilla', 'chocolate', 'chocolate', 'vanilla',
            'chocolate', 'chocolate', 'vanilla', 'chocolate', 'chocolate', 'vanilla', 'chocolate', 'chocolate', 'vanilla', 'chocolate',
            'vanilla', 'chocolate', 'chocolate', 'vanilla', 'chocolate', 'chocolate', 'vanilla', 'chocolate', 'chocolate', 'vanilla']
```

To compute the 95% confidence interval for the population proportion, you can use the **binom.interval** function from the **scipy.stats** library. You need to pass the confidence parameter as 0.95, the number of trials as the sample size, and the probability of success as the sample proportion.

Tutorial 5.15: An example of computing a confidence interval for the proportion of students in a class who prefer chocolate ice cream to vanilla ice cream, using the above list of preferences, is as follows:

```
import scipy.stats as st
n = len(preferences) # sample size
```

```
3. p = preferences.count('chocolate') / n # sample proportion
4. ci = st.binom.interval(confidence=0.95, n=n, p=p) # confidence interval
5. print(f"Confidence interval: {ci}")
```

Output:

```
1. Confidence interval: (26.0, 39.0)
```

This indicates a 95% confidence level that the actual proportion of students in the class who prefer chocolate ice cream over vanilla ice cream falls between 26/50 and 39/50.

Tutorial 5.16: A Python program that calculates the confidence interval for a proportion. In this case, we are estimating the proportion of people who prefer coffee over tea.

For example, if a survey of 100 individuals is conducted and 60 of them express a preference for coffee over tea, the proportion is 0.6. The confidence interval provides a range within which the actual proportion of coffee enthusiasts in the population are likely to fall.

```
1. import scipy.stats as stats
2. n = 100 # Number of trials
3. x = 60 # Number of successes
4. # Calculate the proportion
5. p = x / n
6. # Confidence level
7. confidence_level = 0.95
8. # Calculate the confidence interval
9. ci_low, ci_high = stats.binom.interval(confidence_level, n, p)
10. print(f"The {confidence_level*100}% confidence interval for the proportion is ({ci_low/n}, {ci_high/n})")
```

This program uses the **binom.interval** function from the

scipy.stats module to calculate the confidence interval. The **binom.interval** function returns the endpoints of the confidence interval for the Binomial distribution. The confidence interval is then scaled by n to give the confidence interval for the proportion.

Output:

```
The 95.0% confidence interval for the proportion is (0.5, 0.69)
```

Confidence interval for differences

A confidence interval for the difference is a range of values that likely contains the true difference between two population parameters with a certain level of confidence. This confidence interval type is suitable when there are two independent data samples from two populations, and the parameters of the two populations need to be compared. For example, suppose you want to compare the average heights of male and female students in a class. You randomly select 10 male and 10 female students and measure their heights in centimeters. The following data is obtained:

```
male_heights = [170, 175, 180, 185, 190, 195, 200, 205, 210, 215]
female_heights = [160, 165, 170, 175, 180, 185, 190, 195, 200, 205]
```

To calculate the 95% confidence interval for the difference between population means, use the **ttest_ind** function from the **scipy.stats** library. Pass the two samples as arguments and set the **equal_var** parameter to **False** if you assume that the population variances are not equal. The function returns the t-statistic and the p-value of the test. Use the p-value to calculate the confidence interval.

Tutorial 5.17: An example of calculating the confidence interval for differences between two population means, is

as follows:

```
import numpy as np
import scipy.stats as st
t_stat, p_value = st.ttest_ind(male_heights, female_heights, equal_var=False) # t-test
mean1 = np.mean(male_heights) # sample mean of male heights
mean2 = np.mean(female_heights) # sample mean of female heights
se1 = st.sem(male_heights) # standard error of male heights
se2 = st.sem(female_heights) # standard error of female heights
sed = np.sqrt(se1**2 + se2**2) # standard error of difference
confidence = 0.95
z = st.norm.ppf((1 + confidence) / 2) # z-score for the confidence level
margin_error = z * sed # margin of error
ci = ((mean1 - mean2) - margin_error, (mean1 - mean2) + margin_error) # confidence interval
print(f"Confidence interval: {ci}")
```

Output:

```
Confidence interval: (-3.269018901755597, 23.269018901755597)
```

This means that there is a 95% confidence that the actual difference between the average heights of male and female students in the class falls between -3.27 and 23.27 cm.

Tutorial 5.18: A Python program that calculates the confidence interval for the difference between two population means, Nepalese and Norwegians.

For example, if you measure the average number of hours of television watched per week by 100 Norwegian and 100

Nepalese, the difference between the means plus or minus the variation provides the confidence interval as follows:

```
import numpy as np
import scipy.stats as stats
# Suppose these are your data
norwegian_hours = np.random.normal(loc=10, scale=2, size=100)  # Normally distributed data with mean=10, std dev=2
nepalese_hours = np.random.normal(loc=8, scale=2.5, size=100)  # Normally distributed data with mean=8, std dev=2.5
# Calculate the means
mean_norwegian = np.mean(norwegian_hours)
mean_nepalese = np.mean(nepalese_hours)
# Calculate the standard deviations
std_norwegian = np.std(norwegian_hours, ddof=1)
std_nepalese = np.std(nepalese_hours, ddof=1)
# Calculate the standard error of the difference
sed = np.sqrt(std_norwegian**2 / len(norwegian_hours) + std_nepalese**2 / len(nepalese_hours))
# Confidence level
confidence_level = 0.95
# Calculate the confidence interval
ci_low, ci_high = stats.norm.interval(confidence_level, loc=(mean_norwegian - mean_nepalese), scale=sed)
print(f"The {confidence_level*100}% confidence interval for the difference in means : ({ci_low:.2f}, {ci_high:.2f})")
```

This program uses the **norm.interval** function from the **scipy.stats** module to compute the confidence interval. The **norm.interval** function returns the endpoints of the confidence interval for the normal distribution. The

confidence interval is then used to estimate the range within which the difference in population means is likely to fall.

Output:

```
1. The 95.0% confidence interval for the difference in mea
   ns : (1.44, 2.65)
```

The output is (1.44, 2.65), we can be 95% confident that the true difference in the average number of hours of television watched per week between Norwegians and Nepalese is between 1.44 and 2.65 hours.

Confidence interval estimation for diabetes data

Here we apply the above point and confidence interval estimation in the diabetes dataset. The diabetes dataset contains information on 768 patients, such as their number of pregnancies, glucose level, blood pressure, skin thickness, insulin level, BMI, diabetes pedigree function, age, and outcome (whether they have diabetes or not). The outcome variable is a binary variable, where 0 means no diabetes and 1 means diabetes. The other variables are either numeric or categorical. One way to use point and interval estimation in the diabetes dataset is to estimate the mean and proportion of each variable for the entire population of patients and construct confidence intervals for these estimates. Another way to use point and interval estimates in the diabetes dataset is to compare the mean and proportion of each variable between the two groups of patients, those with diabetes and those without diabetes, and construct confidence intervals for the differences. The implementation is shown in the following tutorials.

Tutorial 5.19: An example to estimate the mean of glucose level for the whole population of patients.

For estimating the mean of glucose level for the whole

population of patients, we can use the sample mean as a point estimate, and construct a 95% confidence interval as an interval estimate as follows:

```
import pandas as pd
import numpy as np
import scipy.stats as st
import matplotlib.pyplot as plt
# Load the diabetes data from a csv file
data = pd.read_csv("/workspaces/ImplementingStatisticsWithPython/data/chapter1/diabetes.csv")
# get the glucose column
x = data["Glucose"]
# get the sample size
n = len(x)
# get the sample mean
mean = x.mean()
# get the sample standard deviation
std = x.std()
# set the confidence level
confidence = 0.95
# get the critical value
z = st.norm.ppf((1 + confidence) / 2)
# get the margin of error
margin_error = z * std / np.sqrt(n)
# get the lower bound of the confidence interval
lower = mean - margin_error
# get the upper bound of the confidence interval
upper = mean + margin_error
print(f"Point estimate of the population mean of glucose level is {mean:.2f}")
print(f"95% confidence interval of the population mean of glucose level is ({lower:.2f}, {upper:.2f})")
```

Output:

```
1. Point estimate of the population mean of glucose level is
 120.89
2. 95% confidence interval of the population mean of gluco
se level is (118.63, 123.16)
```

This means the point estimation is 120.89. And that we are 95% confident that the true mean of glucose level for the whole population of patients is between 118.63 and 123.16. Now, further let us compute the standard error and margin of error of the estimation and see what it shows.

Tutorial 5.20: An implementation to compute the standard error and the margin of error when estimating the mean glucose level for the whole population of patients, is as follows:

```
1. import pandas as pd
2. import numpy as np
3. import scipy.stats as st
4. import matplotlib.pyplot as plt
5. # Load the diabetes data from a csv file
6. data = pd.read_csv("/workspaces/ImplementingStatistic
sWithPython/data/chapter1/diabetes.csv")
7. # get the glucose column
8. x = data["Glucose"]
9. # get the sample size
10. n = len(x)
11. # get the sample mean
12. mean = x.mean()
13. # get the sample standard deviation
14. std = x.std()
15. # set the confidence level
16. confidence = 0.95
17. # get the critical value
```

```
z = st.norm.ppf((1 + confidence) / 2)
# define a function to calculate the standard error
def standard_error(std, n):
    return std / np.sqrt(n)

# define a function to calculate the margin of error
def margin_error(z, se):
    return z * se

# call the functions and print the results
se = standard_error(std, n)
me = margin_error(z, se)
print(f"Standard error of the sample mean is {se:.2f}")
print(f"Margin of error for the 95% confidence interval is {me:.2f}")
```

Output:

```
Standard error of the sample mean is 1.15
Margin of error for the 95% confidence interval is 2.26
```

The average glucose level is estimated with a precision of 1.15 units and a 95% confidence interval of plus or minus 2.26 (i.e, *120.89 ± 2.26*).

Tutorial 5.21: An implementation for estimating the proportion of patients with diabetes for the whole population of patients.

To estimate the proportion of patients with diabetes for the whole population of patients, we can use the sample proportion as a point estimate, and construct a 95% confidence interval as an interval estimate.

```
import pandas as pd
import numpy as np
import scipy.stats as st
import matplotlib.pyplot as plt
```

```
# Load the diabetes data from a csv file
data = pd.read_csv("/workspaces/ImplementingStatisticsWithPython/data/chapter1/diabetes.csv")
# get the outcome column
y = data["Outcome"]
# get the sample size
n = len(y)
# get the sample proportion
p = y.mean()
# set the confidence level
confidence = 0.95
# get the critical value
z = st.norm.ppf((1 + confidence) / 2)
# get the margin of error
margin_error = z * np.sqrt(p * (1 - p) / n)
# get the lower bound of the confidence interval
lower = p - margin_error
# get the upper bound of the confidence interval
upper = p + margin_error
print(f"Point estimate of the population proportion of patients with diabetes is {p:.2f}")
print(f"95% confidence interval of the population proportion of patients with diabetes is ({lower:.2f}, {upper:.2f})")
```

Output:

```
Point estimate of the population proportion of patients with diabetes is 0.35
95% confidence interval of the population proportion of patients with diabetes is (0.32, 0.38)
```

This means the 35% of patient are diabetic and that we are 95% confident that the true proportion of patients with diabetes for the whole population of patients is between

0.32 and 0.38.

Confidence interval estimate in text

We apply point and confidence interval estimation to analyze the word length in transaction narrative notes. This statistical method helps us examine text file data, specifically analyzing the format of the transaction narratives provided.

The narrative contain text in following format.

```
1. Date: 2023-08-01
2. Merchant: VideoStream Plus
3. Amount: $9.99
4. Description: Monthly renewal of VideoStream
   Plus subscription.
5.
6. Your subscription to VideoStream Plus has been
   successfully renewed for $9.99.
```

Tutorial 5.22: An implementation of point and confidence interval in the transaction narrative text to compute the average word and the 95% confidence interval in the text file, is as follows:

```
1. import scipy.stats as st
2. # Read the text file as a string
3. with open("/workspaces/ImplementingStati
   sticsWithPython/data/chapter1/TransactionNarrative/1.
   txt", "r") as f:
4.     text = f.read()
5. # Split the text by whitespace characters and remove e
   mpty strings
6. words = [word for word in text.split() if word]
7. # Calculate the length of each word
8. lengths = [len(word) for word in words]
```

```
# Calculate the point estimate of the mean length
mean = sum(lengths) / len(lengths)
# Calculate the standard error of the mean length
sem = st.sem(lengths)
# Calculate the 95% confidence interval of the mean length
ci = st.t.interval(confidence=0.95, df=len(lengths)-1, loc=mean, scale=sem)
# Print the results
print(f"Point estimate of the mean length is {mean:.2f} characters")
print(
    f"95% confidence interval of the mean length is {ci[0]:.2f} to {ci[1]:.2f} characters")
```

Output:

```
Point estimate of the mean length is 6.27 characters
95% confidence interval of the mean length is 5.17 to 7.37 characters
```

Here, the mean length point estimate is the average length of all the words in the text file. It is a single value that summarizes the data. You calculated it by dividing the sum of the lengths by the number of words. The point estimate of the average length is 6.27 characters. This means that the average word in the text file is about 6 characters long. Similarly, the 95% confidence interval of the mean length is an interval that, with 95% probability, contains the true mean length of the words in the text file. It is a range of values that reflects the uncertainty of the point estimate. You calculated it using the **t.interval** function, which takes as arguments the confidence level, the degrees of freedom, the point estimate, and the standard error of the mean. The standard error of the mean is a measure of how much the point estimate varies from sample to sample. The 95%

confidence interval for the mean is 5.17 to 7.37 characters. This means that you are 95% confident that the true average length of the words in the text file is between 5.17 and 7.37 characters.

Tutorial 5.23: An implementation to visualize computed point and confidence interval in a plot, is as follows:

```
import matplotlib.pyplot as plt
# Create a figure and an axis
fig, ax = plt.subplots()
# Plot the point estimate as a horizontal line
ax.hlines(mean, xmin=0, xmax=len(lengths), color='blue', label='Point estimate')
# Plot the confidence interval as a shaded area
ax.fill_between(x=range(len(lengths)), y1=ci[0], y2=ci[1], color='orange', alpha=0.3, label='95% confidence interval')
# Add some labels and a legend
ax.set_xlabel('Word index')
ax.set_ylabel('Word length')
ax.set_title('Confidence interval of the mean word length')
ax.legend()
# Show the plot
plt.show()
```

Output:

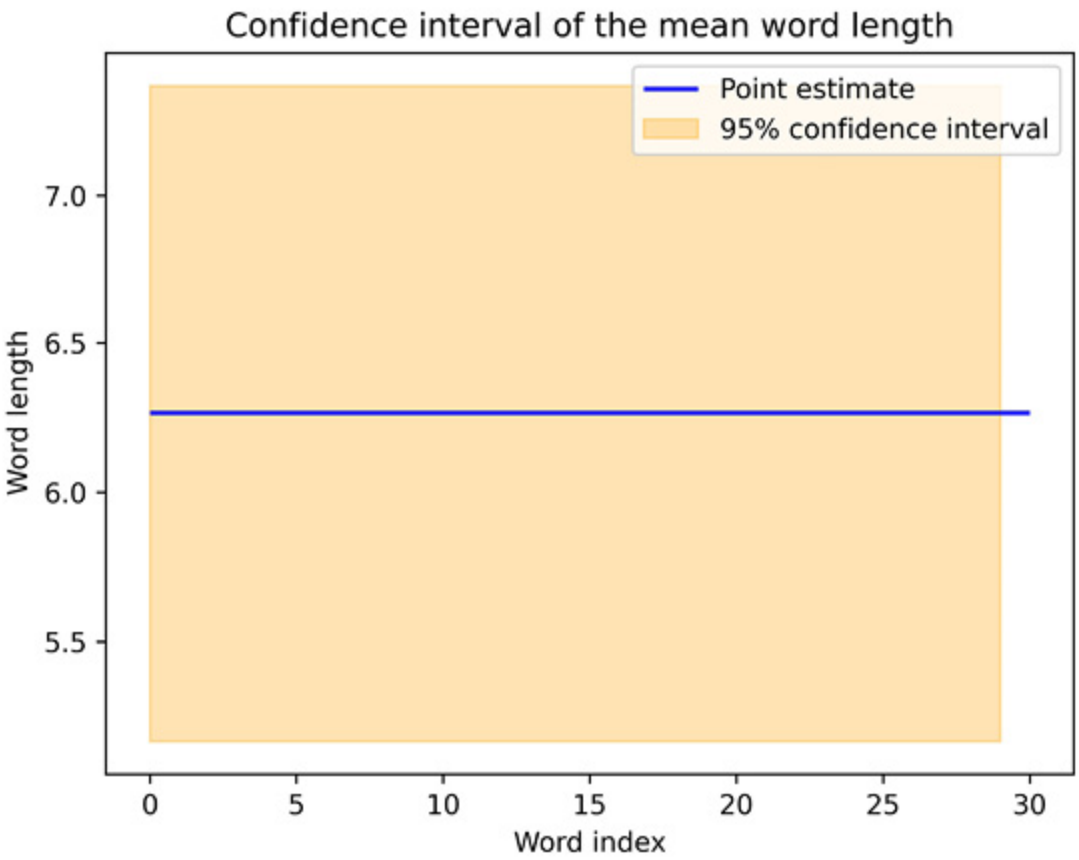

Figure 5.1: *Plot showing point estimate and confidence interval of mean word length*

The plot shows the confidence interval of the mean word length for some data. The plot has a horizontal line in blue representing the point estimate of the mean, and a shaded area in orange representing the 95% confidence interval around the mean.

Conclusion

In this chapter, we have learned how to estimate unknown population parameters from sample data using various methods. We saw that there are two types of estimation: point estimation and interval estimation. Point estimation gives a single value as the best guess for the parameter, while interval estimation gives a range of values that includes the parameter with a certain degree of confidence.

We have also discussed the errors in estimation and how to measure them using standard error and margin of error. In addition, we have shown how to construct and interpret different confidence intervals for different scenarios, such as comparing means, proportions, or correlations. We learned how to use t-tests and p-values to test hypotheses about population parameters based on confidence intervals. We applied the concepts and methods of estimation to real-world examples using the diabetes dataset and the transaction narrative.

Similarly, estimation is a fundamental and useful tool in data analysis because it allows us to make inferences and predictions about a population based on a sample. By using estimation, we can quantify the uncertainty and variability of our estimates and provide a measure of their reliability and accuracy. Estimation also allows us to test hypotheses and draw conclusions about the population parameters of interest. It is used in a wide variety of fields and disciplines, including economics, medicine, engineering, psychology, and the social sciences.

We hope this chapter has helped you understand and apply the concepts and methods of estimation in data analysis. The next chapter will introduce the concept of hypothesis and significance testing.

Join our book’s Discord space

Join the book's Discord Workspace for Latest updates, Offers, Tech happenings around the world, New Release and Sessions with the Authors:

https://discord.bpbonline.com